Trevor Anderson
tba428

12 September 2019

1. In Java, dynamic polymorphism must come from an extended base class. The class shape would be declared, with the draw method included. The circle and square methods would explicitly declare that they extend the class and implement the methods. The benefit to python comes in that if there is a basic behavior anticipated for many objects, those objects can be treated the same and the interpreter will be able to find the methods for each without having to explicitly declare them. Java's benefit is that it provides specific definition, so no unanticipated behavior will occur.
2. See python file for solutions
3. As described in timing_test.ipybn, the &timeit command runs the command a number of times, then returns an object which contains a list of the measurements taken. I then calculated the average time from each of these. Doing this, I looped while increasing the size of random numbers in the array. For convenience, I wrote a method which first increased the array size in large steps, then after hitting the target value, stepped back and then increased in smaller steps to get closer to the actual value.
4. To debug the method, I simply iterated through each 52 times, expecting the deck to eventually be gone and to see every possible output. I added the %debug command to each method call, and then was able to step through and see objects at every step. This made it easy to see the bug in the drawCard method, identified by a comment in my jupyter notebook, and fixed.